(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2004-504662
(P2004-504662A)
(43)公表日　平成16年2月12日(2004.2.12)

(51)Int.Cl.[7]	F I		テーマコード（参考）
G06F 11/00	G06F	9/06 66ON	5B076

審査請求 未請求　予備審査請求 有　（全 54 頁）

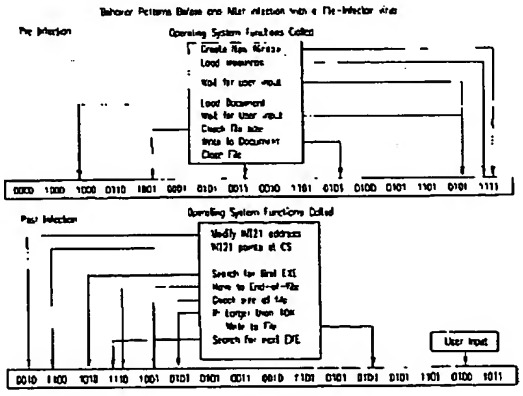
(21)出願番号	特願2002-512775 (P2002-512775)
(86)(22)出願日	平成13年6月14日 (2001.6.14)
(85)翻訳文提出日	平成15年1月14日 (2003.1.14)
(86)国際出願番号	PCT/US2001/019142
(87)国際公開番号	WO2002/006928
(87)国際公開日	平成14年1月24日 (2002.1.24)
(31)優先権主張番号	60/218,489
(32)優先日	平成12年7月14日 (2000.7.14)
(33)優先権主張国	米国 (US)
(31)優先権主張番号	09/642,625
(32)優先日	平成12年8月18日 (2000.8.18)
(33)優先権主張国	米国 (US)

(71)出願人　503020839
ヴィースィーアイエス　インコーポレイテッド
アメリカ合衆国　カリフォルニア州　９４５３９－６６９３　フレモント　リトルフット　プレース　４５４１９
(74)代理人　100083806
弁理士　三好　秀和
(74)代理人　100068342
弁理士　三好　保男
(72)発明者　ヴァン　デア　マデ、　ピーター　エイ．ジェイ．
オーストラリア国　ニューサウスウェールズ州　２１０６　ニューポート　ビーチノール　ストリート　１７

最終頁に続く

(54)【発明の名称】コンピュータ免疫システムおよびコンピュータシステムにおいて望ましくないコードを検出する方法

(57)【要約】
自動分析システムは、コンピュータシステム内においてウイルスおよび他のタイプの悪意あるコードを検出する。この検出は、前記コンピュータシステムに導入した各コンピュータプログラムに対し、挙動パターンを作成しそれを分析することによって行う。挙動パターンの作成は、前記コンピュータシステム内に作成した仮想マシンが行う。前記挙動パターンに対して最初の分析を行い、感染したプログラムを前記コンピュータシステムに初期導入する際に特定する。この分析システムは、挙動パターン、動作順序、および分析結果をデータベースに格納する。新規に感染したプログラムは、そのプログラム用の挙動パターンを新しく作成し、格納した挙動パターンに基づいてそれを分析し、感染パターンまたはペイロードパターンの存在を特定することにより検出できる。
【選択図】図１



【特許請求の範囲】
【請求項１】
中央処理装置の機能とメモリの機能とをシミュレートするソフトウエアからなる仮想マシンを、コンピュータシステム内において初期化し、
対象プログラムを前記仮想マシンにおいて仮想的に実行することにより、前記対象プログラムが前記仮想マシンを介してのみ前記コンピュータシステムと相互作用するようにし、
前記仮想実行した対象プログラムの挙動を分析し、悪意あるコードの挙動の出現を特定し、その悪意あるコードの挙動の出現を挙動パターンに示し、
前記分析処理後に前記仮想マシンを終了することにより、前記仮想マシンが持っていた前記対象プログラムのコピーを前記コンピュータシステムから削除する、コンピュータシステム内のプログラムコード中に存在する悪意あるコードを特定する方法。
【請求項２】
前記仮想マシンは、入出力ポートと、オペレーティングシステムデータ領域と、オペレーティングシステムアプリケーションプログラムインタフェースとの機能をシミュレートする、請求項１記載の方法。
【請求項３】
前記仮想マシンは、仮想ビジュアルベーシックエンジンをさらに備える、請求項２記載の方法。
【請求項４】
前記対象プログラムの仮想実行は、前記対象プログラムと前記シミュレートしたオペレーティングシステムアプリケーションプログラムインタフェースとを相互作用させる、請求項２記載の方法。
【請求項５】
前記対象プログラムは、前記コンピュータシステムに新しく導入されるプログラムであり、仮想実行前には実行されない、請求項１記載の方法。
【請求項６】
前記仮想マシンによって第１プログラムの第１インスタンスを分析し、第１挙動パターンを作成し、前記コンピュータシステム内のデータベースに格納した後、前記方法はさらに、
前記第１プログラムが変更されたことを判定し、
前記変更された第１プログラムを前記仮想マシンにおいて実行して分析し、第２挙動パターンを提供し、
前記第１挙動パターンと前記第２挙動パターンとを比較する、請求項１記載の方法。
【請求項７】
前記第１プログラムが変更されるたびに新しい挙動パターンを作成する、請求項６記載の方法。
【請求項８】
前記第１プログラムの変更中に導入された悪意あるコードは、前記第１挙動パターンと前記第２挙動パターンとを比較することにより検出する、請求項６記載の方法。
【請求項９】
前記変更された第１プログラムが前記第１プログラムの新しいバージョンである場合、前記第１挙動パターンは前記第２挙動パターンと実質的に同等である、請求項６記載の方法。
【請求項１０】
前記挙動パターンは、前記対象プログラムの仮想実行による実行機能を特定し、前記方法はさらに、前記仮想マシンにおいて前記対象プログラムが仮想実行する機能の実行順序を追跡する、請求項１記載の方法。
【請求項１１】
中央処理装置の機能と、メモリの機能と、仮想オペレーティングシステムへの割込み呼び出しを含むオペレーティングシステムの機能とをシミュレートするソフトウエアからなる

仮想マシンをコンピュータシステム内において初期化し、
対象プログラムを前記仮想マシンにおいて仮想的に実行することにより、前記対象プログラムが前記仮想マシンを介して前記仮想オペレーティングシステムおよび前記仮想中央処理装置と相互作用するようにし、
前記仮想実行中の対象プログラムの挙動を監視し、悪意あるコードの存在を特定し、悪意あるコードの挙動の出現を挙動パターンに示し、
前記仮想マシンを終了し、前記分析した対象プログラムの挙動パターン特性の記録を残す、コンピュータシステム内のプログラムコード中に存在する悪意あるコードを特定する方法。

【請求項１２】
前記記録は、前記コンピュータシステム内の挙動レジスタ内にある、請求項１１記載の方法。

【請求項１３】
前記仮想マシンによって第１プログラムの第１インスタンスを分析し、第１挙動パターンを作成し、前記コンピュータシステム内のデータベースに格納した後、前記方法はさらに、
前記第１プログラムが変更されたことを判定し、
前記変更された第１プログラムを前記仮想マシンにおいて実行して分析し、第２挙動パターンを提供し、
前記第１挙動パターンと前記第２挙動パターンとを比較する、請求項１１記載の方法。

【請求項１４】
前記第１プログラムが変更されるたびに新しい挙動パターンを作成する、請求項１３記載の方法。

【請求項１５】
前記第１プログラムの変更中に導入された悪意あるコードは、前記第１挙動パターンと前記第２挙動パターンを比較することにより検出する、請求項１３記載の方法。

【請求項１６】
前記変更された第１プログラムが前記第１プログラムの新しいバージョンである場合、前記第１挙動パターンは前記第２挙動パターンと実質的に同等である、請求項１３記載の方法。

【請求項１７】
前記挙動パターンは、前記対象プログラムの仮想実行による実行機能を特定し、前記方法はさらに、前記仮想マシンにおいて前記対象プログラムが仮想実行する機能の実行順序を追跡する、請求項１３記載の方法。

【発明の詳細な説明】

【０００１】
優先権主張情報
本出願は、２０００年７月１７日出願の米国特許仮出願第６０／２１８，４８９号について優先権を主張し、その全てを参照により組み込む。

【０００２】
発明の背景
１．発明の属する技術分野
コンピュータウイルスやトロイの木馬等、悪意ある挙動または自己増殖を行うコンピュータプログラムの検出に関する。

【０００３】
２．関連する技術の説明
ウイルスの検出は、パーソナルコンピュータ時代の重要事項である。インターネット等の通信ネットワークの発展、データ交換の増加、通信用電子メールの急成長に伴い、通信やファイル交換を介したコンピュータの感染は、ますます重要な問題になっている。感染は様々な形を取るが、代表的にはコンピュータウイルス、トロイの木馬、その他悪意あるコ

ードに関係している。電子メールを介したウイルス攻撃に関する最近の事件は、感染速度および被害の大きさの点で劇的であった。インターネットサービスプロバイダ（ＩＳＰ）や企業は、サービス障害や電子メール機能損失の被害を受けた。多くの場合、ファイル交換や電子メール経由の感染を適切に防止しようとすると、コンピュータ利用者に多大の不便をもたらす。ウイルス攻撃を検出し対処するための進んだ方法が求められている。

【０００４】

従来のウイルス検出方法の１つは、シグネチャスキャンである。シグネチャスキャンシステムは、既知の悪意あるコードから抽出したサンプルコードパターンを使用し、それらパターンを他のプログラムコード内に探す。場合によっては、スキャンするプログラムコードをまずエミュレーションによって解読し、その解読したコード内にシグネチャまたは機能シグネチャを探す。このシグネチャスキャン方法の主たる限界は、既知の悪意あるコードしか検出できないことである。すなわち、記憶している既知の悪意あるコードのシグネチャサンプルに一致するコードしか感染を検出できない。これまでに特定されていないウイルスや悪意あるコード、あるいはシグネチャデータベースの更新後に作られたウイルスや悪意あるコードは、全く検出できない。すなわちこの方法は、新しく発生したウイルスを検出しない。また、既に検出しシグネチャデータベースに格納したシグネチャであっても、それが書き換えられていれば、検出できない。

【０００５】

さらにシグネチャ分析法は、コード内におけるシグネチャが期待する形で整列していないと、ウイルスを特定できない。ウイルスの作者は、オペレーションコードを置き換えることにより、あるいはダミーコードや任意コードをウイルス機能に挿入することにより、ウイルスの特定を難しくしている。意味のないコードを挿入することによりウイルスのシグネチャを変更し、シグネチャスキャンプログラムによる検出を不能にした上で、ウイルスの繁殖や感染を可能にすることも考えられる。

【０００６】

他のウイルス検出方法として、完全性検査がある。完全性検査システムは、既知の健全なアプリケーションプログラムコードからコードサンプルを抽出する。そのコードサンプルと当該プログラムファイルの情報とを一緒に格納する。この情報とは、実行可能プログラムヘッダ、ファイル長さ、サンプルの日付および時間を含む。このデータベースに基づきプログラムファイルを定期的に検査し、プログラムファイルが変更されていないことを確認する。完全性検査プログラムは、ユーザがコンピュータのオペレーティングシステムをアップグレードしたり、アプリケーションソフトウエアのインストールやアップグレードを行うと、長大な変更ファイルリストを作成する。完全性検査に基づくウイルス検出システムの主な欠点は、アプリケーションプログラムのあらゆる変更に対して多数のウイルス活動警告を出すことである。利用者は、コンピュータシステムに対する本当の攻撃を示す警告が出されても、それと判断することが困難である。

【０００７】

チェックサム監視システムは、各プログラムファイルについて巡回冗長検査（ＣＲＣ）値を発生することにより、ウイルスを検出する。ＣＲＣ値の変化によってプログラムファイルの変更を検出する。チェックサム監視が完全性検査よりも優れている点は、悪意あるコードにとってチェックサム監視を破るのはより困難なことである。一方、チェックサム監視は、完全性検査システムと同じく、多くの偽警告を出すこと、どの警告が実際のウイルスあるいは感染を示すのかを特定しにくいことにおいて限界がある。

【０００８】

挙動阻止システムは、対象コンピュータのオペレーティングシステムと連係し、悪意ある挙動の兆候を監視し、ウイルス挙動を検出する。悪意ある挙動を検出すると、それを阻止し、利用者に危険な挙動が行われようとしていることを知らせる。利用者は、悪意の可能性があるコードでもその動作を許可できる。そのため挙動阻止システムは、いくぶん信頼性が低い。なぜならシステムの効果は、利用者の入力に依存するからである。さらに常駐挙動阻止システムは、悪意あるコードがそれを検出し無効にすることもある。

【０００９】
他の感染検出の従来方法は、おとりファイルを使用する。この方法は、一般に別のウイルス検出方法との組合せで使用し、現に活動している感染を検出する。すなわち悪意あるコードは、対象コンピュータにおいて実際に活動しており、ファイルを変更している。ウイルスがおとりファイルを変更した時、それを検出できる。多くのウイルスは、おとりファイルを知っており、非常に小さいファイルや明らかなおとりファイルを変更しない。おとりファイルは、その構造や所定ファイル名から分かる。
【００１０】
ウイルスや他の悪意あるコードを検出するための進んだ技術が求められている。
【００１１】
発明の開示
本発明の一態様は、コンピュータシステム内のプログラムコード中に存在する悪意あるコードを特定する方法を提供する。この方法は、前記コンピュータシステム内において、仮想マシンを初期化することを含む。初期化した前記仮想マシンは、中央処理装置の機能とメモリの機能とをシミュレートするソフトウエアを備える。前記仮想マシンは、対象プログラムを仮想的に実行し、当該対象プログラムは、前記仮想マシンを介してのみ前記コンピュータシステムと相互作用する。前記方法は、前記仮想実行した対象プログラムの挙動を分析し、悪意あるコードの挙動の出現を特定し、その悪意あるコードの挙動の出現を挙動パターンに示す。前記仮想マシンは、前記分析処理後に終了し、前記仮想マシンにあった前記対象プログラムのコピーを前記コンピュータシステムから削除する。
【００１２】
本発明の他の態様は、コンピュータシステム内のプログラムコード中に存在する悪意あるコードを特定する方法を提供する。この方法は、前記コンピュータシステム内において仮想マシンを初期化することを含む。前記仮想マシンは、中央処理装置の機能と、メモリの機能と、仮想オペレーティングシステムへの割込み呼び出しを含むオペレーティングシステムの機能とをシミュレートするソフトウエアを備える。対象プログラムを前記仮想マシン内で仮想的に実行し、前記対象プログラムと前記仮想オペレーティングシステムおよび前記仮想中央処理装置とを前記仮想マシンを介して相互作用させる。仮想実行中の前記対象プログラムの挙動を監視し、悪意あるコードの存在を特定し、悪意あるコードの挙動の出現を挙動パターンに示す。前記仮想マシンを終了し、分析した対象プログラムの挙動パターン特性の記録を残す。
【００１３】
発明を実施するための最良の形態
本発明の好適実施例に基づく自動分析システムは、コンピュータシステム内においてウイルスおよび他の悪意あるコードを検出する。この検出は、対象コンピュータシステムに導入する各プログラムについて挙動パターンを作成し、その挙動パターンを分析することによって行う。新しいコンピュータプログラムまたは変更したコンピュータプログラムは、コンピュータシステムで実行する前に分析する。最も好適な方法として、コンピュータシステムは、当該コンピュータシステムをシミュレートした仮想マシンを始動させる。その仮想マシンは、新しいコンピュータプログラムまたは変更したコンピュータプログラムを実行し、物理的コンピュータシステムがその新しいコンピュータプログラムを実行する前に、そのプログラムの挙動パターンを作成する。その挙動パターンに対して初期分析を行い、プログラムが最初にコンピュータシステムに提供された時、感染プログラムを特定する。この分析システムは、挙動パターンとそれに対応する分析結果とをデータベースに格納する。新たに感染したプログラムの検出は、そのプログラムに対応する格納挙動パターンと新たに作成した挙動パターンとの差分を計算し、その差分を分析し、感染または悪意あるコードに関連するペイロードパターンを特定することにより行う。
【００１４】
プログラミングにおいては、様々な用語を用いて様々な機能的プログラミングサブユニットを記述する。時代によってあるいはプログラミング言語によって、各種サブユニットは

、ファンクション、ルーチン、サブプログラム、サブルーチン等の名前で呼ばれてきた。このような呼び方、状況、差異等は、本発明にとってさほど重要ではない。従って本明細書は、単にプログラムという呼び方を使う。プログラムとは、あらゆるサイズの機能的プログラミングユニットであり、コンピュータシステムまたはコンピューティング環境において、定義したタスクを十分に実行する。さらにワードプロセッサプログラムのマクロ、例えばマイクロソフトワード文書のビジュアルベーシックマクロが実行するような特化した機能も、プログラムである。この意味において、各文書もプログラムと考えることができる。

【００１５】

便宜上および簡潔のため、本明細書は、ウイルスの用語を公知の意味合いで用いる。すなわちウイルスは、自己増殖するプログラムであり、コンピュータシステムには望まれないものである。また本明細書で用いるウインドウズの用語は、マイクロソフト社がウインドウズの商標名で販売するいずれかのパーソナルデスクトップオペレーティングシステムを示す。ＰＣまたはパーソナルコンピュータの用語は、特記しない限り、公知のｘ８６構造に基づくコンピュータシステムを示し、インテル社がペンティアムの商標名で販売するマイクロプロセッサおよびその後継のマイクロプロセッサおよび構造を基本とするコンピュータを含む。本明細書は、本発明の各実施態様を示すことを目的とする。本発明の各態様は、図示したパーソナルコンピュータシステムに加え、様々な範囲のコンピュータシステムに適用可能である。

【００１６】

本発明者は、様々なウイルスおよび他の悪意あるソースコードの挙動を分析してきた。そしてウイルスの一般的な特徴を特定した。ウイルスは、別のプログラムに感染し、それによって別のコンピュータに感染しようとする。従ってウイルスは、感染ループを含み、他の実行可能プログラムに自分自身をコピーする。時には、例えばビジュアルベーシックマクロウイルスのように、文書に自分自身をコピーする。ウイルスおよびトロイの木馬は、一般にペイロードを含む。このペイロードによりウイルスは、感染したシステムを侵し伝染する。ペイロードは、例えばウイルスを知らせるポップアップメッセージであり、感染したコンピュータを破壊する悪意ある機能である。例えば、ハードディスクのデータを破壊したり消去し、バイオス（ＢＩＯＳ）フラッシュメモリやＥＥＰＲＯＭ内のバイオスを変更したり不能にする。

【００１７】

ウイルスの別の共通特徴は、ウイルスがメモリに常駐することである。ＤＯＳウイルスは、自分自身をメモリにコピーしそこに常駐する。ほとんどのウイルスは、明確な常駐終了（ＴＳＲ）呼び出しを用いる代わりに、ウイルスをハイメモリにコピーする手続きを使用する。するとウイルスは、ハイメモリブロック内のデータを直接変更できる。この感染方法の別の側面は、割り込みベクトルを変更し、メモリ常駐ウイルスあるいは悪意ある手続きが変更したメモリブロックを指示することである。これら変更されたメモリブロックは、感染手続きを記憶する。ウインドウズ限定のウイルスは、自分自身を強制的にリング０に設定する。これを例えばコールゲートやＤＰＭＩ呼び出しを用いて行い、システムトレイ等のシステムユーティリティに常駐する。

【００１８】

このような挙動は、ウイルスの特徴であり、一般に他の善意のプログラムには見られない。従って、あるプログラムが前記挙動のいずれか、あるいはそれらのいくつか、あるいはそれら全てを持っていれば、そのプログラムをウイルスあるいはウイルスに感染していると特定できる。本発明の好適実施例において、これら挙動の出現あるいはそれら挙動の組合せの出現は、感染したプログラムの挙動特性を表す挙動パターンデータにおいて、ビットの集合として示す。正常なファイルおよび感染したファイルの挙動パターン例を図１に示す。

【００１９】

本発明の好適実施例において、新しくロードしたあるいは呼び出したプログラムの挙動は

、仮想マシンで分析する。この仮想マシンは、完全なＰＣあるいは十分に完全なＰＣをソフトウエアでシミュレートしたものであり、プログラムの挙動パターンを作成する。仮想ＰＣは、新しいプログラムあるいは変更のあったプログラムの実行をシミュレートし、システム機能の一部をシミュレートし、疑わしいプログラムの挙動を監視し、その挙動の記録を作成する。この記録を分析することにより、対象プログラムがウイルスあるいは悪意ある挙動を示したかを判定する。仮想マシンによる仮想実行の結果は、前記新しいプログラムの挙動パターンとなる。詳細は後述するが、仮想ＰＣが作成した挙動パターンは、プログラムがウイルスに感染しているか、あるいはそれ自体がウイルスかを特定する。仮想実行を使用しての新しいプログラムに対するウイルス分析の利点は、仮想マシンは仮想であるから、仮想実行した新しいプログラムがウイルスを含んでいても、仮想マシンが感染するだけなことである。仮想マシンにおいて感染したインスタンスは、シミュレーション後に削除する。従って感染は不完全であり、ウイルスは伝染しない。仮想マシンを削除しても挙動パターンは残り、それを使って分析プログラムがウイルスの存在および新しいプログラム内の感染を特定する。

【００２０】

最も好適であるのは、新しいプログラムを分析する毎に仮想マシンの新しいインスタンスを作成することにより、以前に分析したウイルスや、その他すべての以前に仮想化したプログラムの影響を排除することである。新しいプログラムは、仮想マシンの新しいインスタンスで実行する。この新しいインスタンスは、詳細を後述するように、変更した割込み呼び出し手順を開始する。仮想マシンは、変更した割込み呼び出し手順と協力して新しいプログラムを実行する。その実行中に仮想マシンは、全てのシステム呼び出し、ＤＰＭＩ／ＤＯＳ割込み、入出力ポート読み出し／書き込み（Ｒ／Ｗ）挙動を監視し、それら挙動に基づき挙動パターンレジスタのビットを設定する。これら挙動パターンのビットは、シミュレーション完了後および仮想ＰＣ終了後も保管する。挙動パターンレジスタに記憶したビットは、挙動パターンであり、仮想実行したプログラムがウイルスあるいは他の悪意あるコードの存在を示す挙動を含むか否かを表す。

【００２１】

前記変更した割込み呼び出し手順は、分析対象プログラムが仮想ＰＣ内において変更した割込みを呼び出し、それら割込みサービスルーチンの各々について挙動パターンを作成する。これにより本発明の好適実施例は、特殊タイプのウイルスも特定できる。この種のウイルスは、最初に割込みサービスルーチンだけを変更し、変更した割込みが他のプログラムによって呼び出された時、始めて伝染する。本発明の実施例は、仮想マシン内において様々な割込みサービスルーチンの変更を許可した後、その変更された割込みを分析することにより、前記遅延伝染機構を検出できる。

【００２２】

別の好適実施例は、挙動パターンの静的な最終版だけを分析する。挙動パターンレジスタのビットが設定される順番を監視することも可能であり、環境によってはそうすることが望ましい。挙動パターンビットの設定順序は、追加情報を提供し、さらなるウイルス挙動の特定を可能にする。挙動パターンビットの設定順序を追跡することは、仮想マシン内で遂行できる。

【００２３】

挙動分析法（ＡＢＭ）の実施例は、変更された新しい未知のあるいは疑わしいプログラムから挙動パターンおよび順序を抽出する。この挙動パターンを使用し、未知プログラムの挙動を分析し、未知プログラムの挙動が悪意あるか否かを判定する。この悪意ある挙動の特定方法は、ホストコンピュータシステムが感染する前に、ウイルス汚染ファイルの特定を可能にする。挙動パターンをデータベースに格納し、変更されたプログラムの挙動を分析し、それが疑わしい（悪意ある）方法で変更されたか否かを判定できる。これは感染後分析を提供する。

【００２４】

前記挙動分析法が従来のウイルス検出法と異なる点は、シグネチャスキャン法や完全性検

査法のようにプログラムコードと記憶パターンとの照合を行わないことである。それに代えて本発明方法は、仮想マシンを使用し、挙動パターンおよび順序を作成する。作成した挙動パターンは、バージョン更新間ではそれほど変化しない。ところがウイルスに感染すると、大幅に変化する。例えばワードプロセッサは、そのプログラムが新しいバージョンのプログラムに置き換えられたり更新されたりしても、やはりワードプロセッサの動作を行う。しかしながらウイルスに感染すると、ワードプロセッサは極端に変化する。その差異は、図１に示すように、挙動パターンに反映する。ワードプロセッサは、ファイル感染コンピュータウイルスに汚染されると、実行可能ファイルを開き、その中にウイルスコードを挿入し、さらに別のファイルを汚染する。これは、図示の挙動パターンにはっきりと反映している。

【００２５】

本発明の好適実施例における分析手順が特に対象とする感染方法は、他の実行可能ファイルまたは文書へのコード挿入、送信または格納しようとする他のアプリケーションへのコード送出、ハイメモリブロックへのコード挿入、およびメモリ制御ブロックの変更である。ただしこれらに限定するものではない。さらに好適実施例に基づく分析法は、破壊的内容を検出する。破壊的内容とは、ディスク領域やバイオス（ＢＩＯＳ）ＲＯＭを書き換えたり、ファイルやディレクトリを削除する機能である。ただしこれらに限定するものではない。本発明の好適実施例における分析法は、ある種のプログラムについてその挙動特性が開発ツールまたはソフトウエアデバッギングツールに特有の特性を示す場合、そのプログラムを例外とみなし、汚染していると特定しない。この種のプログラムにとって、変更挙動は不可欠の通常機能の一部である。開発ツールがウイルスに感染すると、そのツールの標準機能の一部ではない機能を示す。すなわち開発処理の一部ではない処理を行うので、その感染を検出できる。この分析において、挙動パターンにおける有効フラグ（１）、無効フラグ（０）、動作実行順序は重要である。

【００２６】

本発明の好適実施例に基づき、仮想マシンまたは仮想ＰＣは、完全なコンピュータシステムをシミュレートする。完全なコンピュータシステムは、エミュレートした中央処理装置（ＣＰＵ）、エミュレートしたメモリ、入出力（Ｉ／Ｏ）ポート、バイオスファームウエア、オペレーティングシステム、およびオペレーティングシステムデータ領域を含むことが好ましい。これは、処理装置の処理だけを単純にエミュレートするのとは対照的である。好適実施例におけるエミュレーションは、プログラム命令を元の形式から命令ストリームに変換し、その命令ストリームは、異なるハードウエアプラットフォームにおいて同一機能を実行する。いくつかのシグネチャスキャンソフトウエアは、疑わしいプログラムのシグネチャをスキャンする前に、エミュレーションによってそのプログラム本体を解読する。一方、仮想化は、オペレーティングシステム呼び出しを含むコンピュータ全体をシミュレートする。これらオペレーティングシステム呼び出しは実際に実行するのではないが、呼び出し元のプログラムにとっては、要求機能を実行しているように見え、実際に実行したかのように正しい値を返す。

【００２７】

前記したように、仮想ＰＣは、ＣＰＵと、メモリと、入出力ポートと、プログラムローダと、オペレーティングシステムアプリケーションプログラムインタフェース（ＡＰＩ）エントリポイントおよびインタフェースとを含む。このように完全な仮想ＰＣを挙動分析法に使用することは、特に好ましい。オペレーティングシステムＡＰＩへの精妙な直接呼び出しを含む仮想化したプログラムの高レベル制御を提供するからである。仮想化したプログラムは、物理的マシン各部へのアクセスを許されないため、ウイルス等の悪意あるコードは、制御環境を抜け出せず、ホストコンピュータシステムに感染する恐れがない。

【００２８】

図２は、好適な挙動分析構成の概略を示す。この図は、仮想マシンとホストコンピュータシステムの構成部品との関係を示す。プログラムコードは、挙動分析エンジンと分析システムとへ送る。これは、入出力ポートビットを操作し、オペレーティングシステムのファ

イルシステムへフッキングしてハードディスクに直接アクセスするか、あるいは前記ハードディスクを順次スキャンすることにより行う。前記プログラムコードが「既知」のファイルか否かをデータベースに基づき検査する。そのファイルが新しいか変更されていれば、それを処理する。その結果の挙動シグネチャを分析あるいは比較し、格納する。分析の結果、ファイルが悪意あるコードを含んでいると分かれば、ウイルス警告を返す。挙動分析法は、（１）ファイル構造抽出、（２）変更検出、（３）仮想化、（４）分析、（５）判定、を含むことが好ましい。

【００２９】

プログラムを仮想化するには、対象プログラムを含むファイルフォーマットを評価しなければならない。エントリポイントコードを抽出し、仮想コンピュータメモリのシミュレートした正しいオフセットにロードしなければならない。この機能は、物理的コンピュータの場合、プログラムローダ機能が行う。プログラムローダ機能は、オペレーティングシステムの一部である。オペレーティングシステムは、様々なファイルフォーマットのプログラムを実行できる。これらファイルフォーマットの例を以下に示す。

【００３０】

【表１】

ＤＯＳ１．０および／またはＣＰ／ＭＣＯＭ	バイナリイメージファイル。メモリの１００ｈへロード。最大サイズ６４Ｋ
ＤＯＳ２．０～ＤＯＳ７．１ＥＸＥ	ＭＺタイプ実行可能フォーマット。ヘッダがロードアドレスのＣＳ：ＩＰを決定。
ウインドウズ３．０実行可能フォーマット	ＮＥタイプ実行可能フォーマット。ＤＯＳコード領域をポイントするＤＯＳＭＺヘッダとウインドウズ（プロテクトモード）コードのエントリポイントを含む新実行可能（ＮＥ）ヘッダとを含む。ＮＥファイルはセグメント化されている。
ＯＳ／２実行可能フォーマット	ＬＥ／ＬＸタイプ実行可能フォーマットは、ＤＯＳＭＺヘッダと、ＤＯＳコード領域と、ＤＯＳコードセグメントに続くＬＥヘッダが決定するプロテクトモードセクションとを含む。リニア実行可能（ＬＥ）ファイルは、ウインドウズ３がシステムユーティリティおよびデバイスドライバ用に使用する。ＬＥファイルはセグメント化されている。ＬＸファイルは、ページテーブルの格納方法に少しの差異を含み、ＯＳ／２オペレーティングシステム用を意図している。ＬＥファイルはセグメント化され、セグメントはページ化されている。
３２ビット実行可能フォーマット	ＰＥタイプ実行可能フォーマットは、ＤＯＳＭＺヘッダと、ＤＯＳコード領域と、ポータブル実行可能ヘッダとを含む。ポータブル実行可能ヘッダは、エントリポイントとプロテクトモードコードのファイルオフセットとを含む。
ＯＬＥ複合ファイル	ＯＬＥ複合ファイル（ＣＯＭ）は、文書ファイルであり、一般にマクロと呼ぶ実行可能フォーマットストリームを含むことができる。全てのオフィス部品は、インターネットエクスプローラバージョン４および５と同様、アプリケーション用ビジュアルベーシックを内蔵する。ウインドウズ９８システムは、ビジュアルベーシックコードをスクリプトファイルから直接実行できる。ビジュアルベーシックコードは、コンパイルされてストリームとして格納される。そしてファイルヘッダのリンクリストに格納したオフセット参照に基づき、ページ化される。
バイナリイメージ	バイナリイメージは、ブートセクタ、マスターブート、パーティションテーブル用である。ブートセクタとＭＢＲは、実行可能コードを含む。この実行可能コードは、起動処理中にメモリの０：７Ｃ００にロードされる。
ドライバファイル	システムドライバは、ヘッダを有するバイナリイメージとして格納する。このヘッダは、ファイルに格納したドライバに関する情報を含む。複数のドライバを同一ファイルに格納できる。

【００３１】
仮想コンピュータローダ機能は、上記ファイルフォーマットおよびバイナリイメージファイルを処理可能である。ローダ機能は、オペレーティングシステムプログラムローダを仮想化することによって実現するため、ホストコンピュータにおいて使用するオペレーティングシステムに依存して変化する。ファイル構造分析手順は、ファイルヘッダおよびファイル構造を調べ、ファイルフォーマットを決定する。この時、ファイル拡張子を使用しな

い。これはファイル拡張子は、一般使用において信頼できないからである。従って前記の「．ＥＸＥ」フォーマットは、ＤＬＬ、ＡＸ、ＯＣＸ、および他の実行可能ファイルフォーマット拡張子を含む。

【００３２】

複合文書ファイルは、ビジュアルベーシックコードやマクロのような実行可能ストリームを含むことができる。複合文書ファイルの構造を図３に示す。複合文書ファイルのヘッダは、リンクリスト（またはファイルアロケーションテーブル）を含む。これは、ディレクトリ構造内で参照され、リンクリストのエントリポイントを示す。リンクリストの各エントリは、次のエントリおよびファイルオフセットを参照する。ストリームはブロックの外に存在し、ファイル内のあらゆる場所にあらゆる順序で散在する。本発明の好適実施例において、複合文書ファイルから抽出したコードは、ビジュアルベーシック逆コンパイラを通してから、ビジュアルベーシックエミュレータへ渡す。全ての複合文書ファイルがコンパイルしたビジュアルベーシックコードを含むわけではない。ハイパーテキストマークアップ言語（ＨＴＭＬ）およびビジュアルベーシックスクリプト（ＶＢＳ）は、ビジュアルベーシックスクリプトコードをテキストとして含むことができる。このコードを抽出し、仮想マシン内においてビジュアルベーシックストリームとして扱うことが好ましい。

【００３３】

ＮＥ、ＰＥ、ＬＥ実行可能ファイルフォーマットは、複雑さにおいて同様だがリンクリストを使用しない。その代わり、これらファイルフォーマットは、セグメントテーブルまたはページテーブルを使用する。ＰＥファイルフォーマットは、ＣＯＦＦファイル仕様に基づく。図４は、これらファイルフォーマットがどのようにして、本発明の一実施例に基づく好適仮想ＰＣとインタフェースするかを示す。好適仮想ＰＣと特定ファイルとのインタフェース方法を評価することにより、ファイルローダは、そのファイルが文書ファイルかバイナリファイルかを判定できる。

【００３４】

ファイルフォーマットを評価しエントリポイントファイルオフセットを計算した後、そのファイルを開く。そして仮想マシンは、関連するコードをデータストリームとしてメモリに読み込む。そのコードの長さは、当該ファイルのヘッダのフィールドから計算する。この情報は、仮想プログラムローダへ送る。仮想プログラムローダは、ファイルヘッダの情報を使用し、抽出したコードを仮想メモリアレイのシミュレートした正しいオフセットへロードする。

【００３５】

メモリマッピングユーティリティは、仮想メモリマップを仮想化するファイルタイプ用のオフセットへマップする。

【００３６】

【表２】

ＤＯＳ（ＣＰ／Ｍ）バイナリイメージファイル（．ＣＯＭ）	オフセットＣＳ：１００ｈ
ＤＯＳ（２．０以上）実行可能フォーマットファイル（ＭＺ－ＥＸＥ）	ヘッダからのオフセットＣＳ：ＩＰ
ウインドウズＮＥ、ＰＥ、ＬＥ	ヘッダからのオフセットＣ００００００＋ＣＳ：ＩＰ
バイナリイメージＭＢＲ、ブートセクタコード	オフセット０：７Ｃ００ｈ
文書ＣＯＭファイル、ＨＴＭＬおよびＶＢＳファイル	特定のオフセット無し。ＶＢＡコード

【００３７】

ローダユーティリティは、プログラムが仮想化されるたびに、物理メモリを動的に仮想コンピュータメモリアレイへ割り当て、新しい仮想マシンの構築を始める。各仮想マシンは、バイオス（ＢＩＯＳ）データ領域、満たされた環境ストリング領域、ＤＯＳデータ領域、メモリ制御ブロック、プログラムセグメントプレフィクス領域、割り込みベクトルテーブル、およびデスクリプタテーブルを含む。仮想マシンの最終的な構造は、仮想化するプログラムのタイプに依存する。従って各仮想化プログラムは、そのプログラムを仮想ＰＣへロードした時に作成される新しいメモリ領域で実行する。従って前のインスタンスが感染プログラムを仮想化したとしても、それは後続のプログラムの挙動に影響しない。仮想マシンは、仮想化プログラムが終了した時に停止し、そのメモリリソースを解放する。そして仮想マシンは、仮想化対象の挙動パターンを完成させる。

【００３８】
図５は、ＣＯＭバイナリファイルおよびＤＯＳプログラム（ＭＺ－ＥＸＥ）ファイルに対する仮想メモリの構成状態を示す。メモリマップおよびマッパユーティリティは、ファイルタイプに応じて調整される。

【００３９】
プログラムローダは、オペレーティングシステムのローダ機能をシミュレートし、物理的コンピュータ内のシステム領域と同等のシステム領域を作成する。これは特に好都合な機能である。なぜなら評価中のコードは、物理的コンピュータシステムで実行しているかのように動作するからである。仮想化したプログラムは、仮想メモリアレイからプリフェッチ命令待ち行列へ命令を取り出すことによって実行する。この待ち行列中の命令を復号し、その動作パラメータからその長さを決定する。

【００４０】
それに従って命令ポインタをインクリメントし、命令ローダが次の命令を取り出せるようにする。仮想マシンは、命令パラメータのｒ／ｍフィールドから、命令の動作に関連するデータを取り出す位置を決定する。データ取り出し機構は、そのデータを取り出し、それを論理ユニットへ送る。その論理ユニットは、当該命令コードが示す動作を行う。処理済みデータの行き先は、命令コードのパラメータから決定する。データ書き込み機構を使用し、前記処理済みデータを、エミュレーションしたメモリまたはエミュレーションしたプロセッサレジスタセットへ書き込む。この処理は、物理的ＣＰＵ（中央処理装置）が実行する処理を正確に反映している。

【００４１】
この処理全体は、図６に示すようにシミュレートする。メモリは４００キロバイト要素のアレイとして存在し、その中でメモリマッピング機構は全てのメモリアクセスをマップする。このメモリ領域のサイズは、将来の実施においてより大きなプログラムを収容するため、調整することができる。ビデオディスプレイは、システムの観点から１２８キロバイトのメモリとしてシミュレートし、仮想コンピュータのメモリマップのＡ０００：０からＢＦＦＦ：Ｆ（含む）までの間にマップする。標準ＩＢＭＰＣ入出力領域は、入出力ポート０～３ＦＦｈを代表する１０２４バイトのアレイとしてシミュレートする。ＣＰＵは、ハイレベルソフトウエア内において、物理的ＣＰＵと同じローレベル機能を実行することによってシミュレートする。

【００４２】
オペレーティングシステムは、７００ｈバイトのメモリアレイ内の領域として実現する。この領域は、バイオス（ＢＩＯＳ）データフィールドと、ＤＯＳデータ領域と、メモリ制御ブロックと、ＤＯＳデバイスとを含む。割り込みベクトルテーブルは、物理的ＰＣにおける場合と同様、メモリ領域の最初の１０２４（４００ｈ）に位置する。ＤＯＳ割込み構造は、正しい値を返すシミュレートした機能として実現し、ＤＯＳ機能をシミュレートした場合に期待される正しい値でメモリアレイを埋めることによって実現する。

【００４３】
オペレーティングシステムは、仮想ＡＰＩ（ＶＡＰＩ）として実現し、全オペレーティングシステムＡＰＩが戻す結果をシミュレートする。

【００４４】
仮想化処理中において、各機能の仮想化に伴い、挙動パターン（Ｔｓｔｒｕｃｔ）フィールドの対応するフラグを設定する。これら機能を呼び出す順番は、シーケンサに記録する。従って挙動パターンは、評価中のプログラムの物理的ＰＣ環境における挙動と正確に一致する。シミュレートした割り込みベクトルであって仮想化プログラムの実行処理中に変化したものは、プログラム仮想化の終了後にそれらを呼び出すことにより、物理的コンピュータにおいてアプリケーションが当該割り込みベクトルをその変更後に呼び出すのと同じことになる。
【００４５】
この機能を説明するため、挙動分析法における各動作を考察する。
【００４６】
当該ディレクトリ内に最初のＥＸＥファイルを検索する；ＦｉｎｄＦｉｒｓｔフラグ設定（Ｔｓｔｒｕｃｔ構造）
これはＰＥタイプの実行可能ファイルか（ヘッダ検査）？；ＥＸＥｃｈｅｃｋフラグ設定
そうでなければＦＡＲジャンプする
それ以外であれば：実行可能ファイルを開き；ＥＸＥａｃｃｅｓｓフラグ設定
セクションテーブルに書き込み；ＥＸＥｗｒｉｔｅフラグ設定
エンドオブファイルを検索し；ＥＸＥｅｏｆフラグ設定
ファイルに書き込み；ＥＸＥｗｒｉｔｅフラグ設定
ファイルを閉じる
次のＥＸＥファイルを検索する；ＥＸＥＦｉｎｄＮｅｘｔフラグ設定
Ｂｉｔ＋１　６４…　…　…　…　…　…　…　…１
戻り：００１０　０１００　１０１０　１０１０　１００１　０１０１　００１０　１１　１１　００１０　１０１０　００１０　０１００　０１００　１００１　０００００　０１０１
値：２　４　Ａ　Ａ　９　５　２　Ｆ　２　Ａ　２　４　４　９　０　５
順序：２１，２２，２３，２４，２６，２９，３Ｅ，１，３６，３８，３Ｂ，３，９，Ｃ，Ｆ，１３，１６，１Ａ，１Ｃ，１Ｅ，２Ｂ，２Ｄ，３０，３２，３４
結果的な挙動パターン：２４ＡＡ９５２Ｆ２Ａ２４４９０５
この挙動パターンが含むフラグは、ユーザがユーザ入力を介してこの処理と対話する機会を持たなかったことを示している（ｕｓｅｒＩｎｐｕｔフラグが設定されていない）。上記順序は、ビットを設定した順番であり、感染順序を特定する。従って上記例の挙動は、ほぼウイルスである。
【００４７】
多くのウイルスは、暗号化されているか、多形であるか、「トリック」を用いてシグネチャスキャナによる検出を避ける。そのような「トリック」を用いる場合、挙動パターンは、より明確にウイルスである傾向を示す。なぜなら、そのようなトリックは通常のアプリケーションは使わないからである。いずれにしても本発明の好適実施例は、ウイルス警告を発生する際に感染手順を考慮するため、偽の警告を避けられる。暗号化ウイルスも検出できる。なぜなら、仮想マシン内でのコードの実行は、物理的ＰＣ環境と同じように、挙動パターンを作成し、暗号化ウイルスまたは多形ウイルスを有効に復号する。好適実施例は、コンピュータの全てを仮想化するので、仮想化したプログラムは決して物理的コンピュータと対話しない。従ってウイルスコードは、仮想マシンから脱出できず、物理的コンピュータに感染できない。
【００４８】
変化検出モジュールは、既存ファイルを６段階において比較し、以前にそのファイルを分析したかを判定する。
【００４９】
・ファイルは同一（エントリーコードポイント、サンプル、ファイル名、およびファイルサイズが同一）である。

【００５０】
・ファイルはデータベースにない（新しいファイル）。
【００５１】
・挙動パターンが記憶パターンと一致する。
【００５２】
・ファイルのエントリコードが変更されている。挙動パターンを以前格納したパターンから２値的に減算する。その結果としてのビットパターンを分析する。
【００５３】
・ファイルのエントリコード、ＣＲＣ、およびヘッダフィールドは同一であるが、ファイル名が変更されている。他のフィールドは変更されていない。
【００５４】
・ファイルの挙動パターンがデータベースにあり、既知のウイルス挙動パターンと一致する。
【００５５】
・ファイルの挙動パターンがデータベースにあり、既知の健全な挙動パターンと一致する。
【００５６】
ファイルの実行可能部分が変更されていれば、そのプログラムを仮想化する。ファイルの実行可能コードが変更されていなければ、そのファイルは、元ファイルが感染していない限り、ウイルスを含むことはない。元ファイルが感染しているなら、そのウイルスを以前の分析で検出しているはずである。既存のプログラムをアップデートした場合、その機能は同一であり、従ってその挙動パターンは格納挙動パターンとほぼ一致する。ビットの変更が感染手順の追加を示せば、そのファイルは感染していると見なす。
【００５７】
二つの検出機構を説明する。これらは並列動作し、いずれも前記挙動パターンを利用する。
【００５８】
感染前検出
これは最も望ましい。感染前検出は、挙動パターンを分析し、システムに導入した新しいまたは変更されたプログラムがウイルス動作を表すことを見つける。検査中のプログラムファイルは、ウイルスを削除することによって修復できる。あるいは、ウイルス感染の除去が極めて困難な場合、あるいは元のコードが書き換えられている場合、そのプログラムファイルを消去できる。感染したプログラムは、この時点でまだ物理的ＰＣ上で実行していないから、ウイルス発見後、物理的ＰＣを修復する必要はない。
【００５９】
感染後検出
感染後検出は、感染前検出が初期感染を見逃した場合に実行する。感染前検出がウイルスを見逃すのは、そのウイルスが最初の実行においてウイルス機能を全く実行せず、感染したルーチンを指す割り込みベクトルを変更しない場合である。これは、いわゆる遅延感染ウイルスおよび同様にふるまう悪意あるコードである。感染後検出は、ウイルスがＰＣ上の最初の実行可能ファイルに感染しようと試みる瞬間にそれを捕捉する。ファイルフック機構は、この実行可能ファイル（文書を含む）への変更の試みを検出する。次に挙動分析エンジンは、最初の実行可能プログラムを分析し、その挙動パターンがウイルス活動を示すように変化していることを発見する。
【００６０】
データベース構造
【表３】

ファイルＩＤ領域	挙動パターン、プログラム名、ファイルサイズ、およびパス
修復構造	ヘッダフィールド、セクションテーブル、およびリロケーションテーブル
セグメントテーブル	セクションテーブルにおける各セクションのサイズおよびオフセット（ウインドウズプログラムのみ）

【００６１】
文書内のマクロウイルスは、実行可能プログラムのように扱う。元のビジュアルベーシックコードは、ビジュアルベーシック文書（ＣＯＭ）ストリームの解読（可能な場合）および逆コンパイリングによって修復する。その結果としてのソースコードは、保存あるいは表示しない。それによって正当なビジュアルベーシックソフトウエアのオリジナル作者の権利を保護する。仮想化した後、ソースコードを破棄する。

【００６２】
前記ウイルス検出システムの１つの欠点は、その初期分析がパターンスキャニングより遅いことである。この欠点は、当該システムの長所が補って余りある。ファイルシステムフッキング手段を使うことにより、全ての新しいファイルは、バックグラウンドにおいて迅速に報告され分析される。すなわちコンピュータからウイルスを無くした後は、完全スキャンの再実行は一般に必要なくなる。ただし新しいプログラムをインストールした時に保護システムを停止していた場合は、別である。シグネチャスキャンに基づく保護システムの場合、コンピュータは、ウイルスシグネチャデータベースを更新するたびに、完全な再スキャンを行わねばならない。ユーザがディスクスキャンを開始した時、変更されていないファイルを再び仮想化することはないから、本発明処理は、少なくともパターンスキャニングと同じくらいに速く、安全性はより高い。記憶した情報は、ファイルやシステム領域へのウイルス被害を修復する助けとなる。従って、ほとんどの場合、完全なあるいは実質的に完全な修復を保証できる。

【００６３】
挙動分析システムの試験的実施において、既知のウイルス技術を１００％検出し、その内訳は感染前検出（９６％）および感染後検出（４％）であった。このテストには、新しいウイルス、変更したウイルス、および既知のウイルスの組合せを使用した。他のウイルス検出方法は、既知ウイルスの１００％を検出し、新しいウイルス、変更したウイルス、および未知のウイルスの検出は０％であった。シグネチャスキャンに基づいた製品については、正確な数字を引用できない。このような製品の結果は、既知ウイルス、変更ウイルス、新ウイルス、および未知ウイルスの混合であり、例えばテストしたウイルスセットの３０％が新ウイルス、変更ウイルス、および未知ウイルスであれば、最終結果として約３０％のウイルスを見逃した。本発明の好適実施例は、そのような関係を全く示しておらず、ウイルスの組合せを変更しても検出効率はほとんど変化しない。

【００６４】
好適実施例を参照して本発明を説明してきた。当業者には明らかな通り、本発明は、現在の好適実施例に制限されるものではない。本発明の範囲を逸脱することなく、これら実施例には様々な変更や拡張が可能である。従って本発明は、前記実施例のいずれによっても制限を受けず、請求の範囲によって定義されるものである。

【図面の簡単な説明】
【図１】挙動分析法に基づき作成した挙動パターンを示す図である。コンピュータウイルスに感染していないコードの挙動パターンと、それに感染したコードの挙動パターンとを示す。各ビットは、挙動を示すフラグと考えることができる。ビットストリーム全体が示

す値は、プログラム挙動の指標である。

【図２】挙動分析法の好適実施例の各部を示すブロック図である。

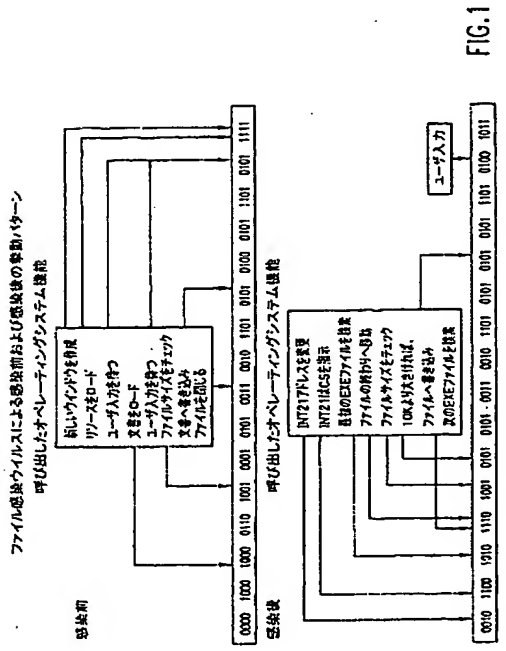
【図３】プログラム構造抽出部およびプログラムローダの機能を示す例として、ＣＯＭファイルフォーマットを示す概略図である。

【図４】各種プログラムファイルフォーマットに対する仮想ＰＣのインタフェースを示す図である。仮想化を実行する前に、プログラムローダは、正しいエントリポイントコードと初期化したデータとをプログラムファイルから抽出することが好ましい。前記エントリポイントコードへのファイルオフセットは、プログラムヘッダに与えられており、前記プログラムファイルのタイプによって変化する。

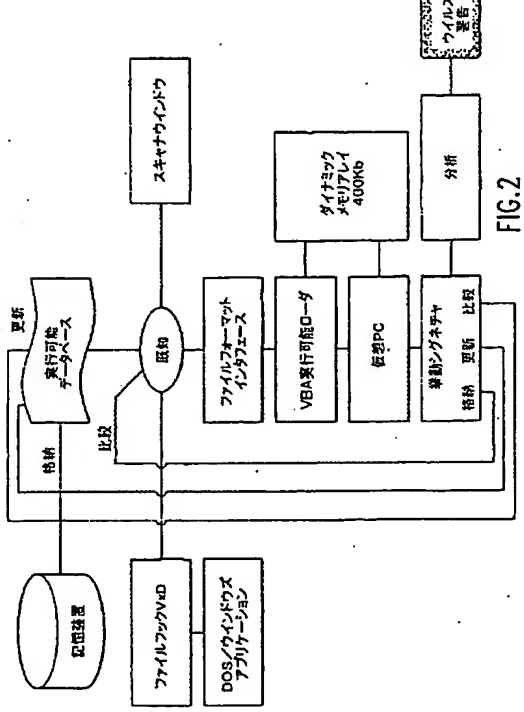
【図５】バイナリイメージ（．ＣＯＭ）プログラムロード後およびＭＺ実行可能プログラムロード後の仮想ＰＣメモリマップを示す概略図である。希望の方法でコードを仮想化するため、仮想ＰＣの構造およびそのメモリマップは、物理的ＰＣでそのコードを実行した場合と同じ情報を含む。物理的ＰＣとは、当該仮想ＰＣを含む仮想マシンを実行するＰＣである。

【図６】好適実施例における仮想ＰＣの各構成部を示す詳細図である。この仮想ＰＣの各構成部は、物理的コンピュータのものと同じである。ただし全ての仮想ＰＣ構成部は、仮想マシンを実行するソフトウエアによって物理的コンピュータ上にシミュレートしたものである。

【図１】



【図２】



【図３】

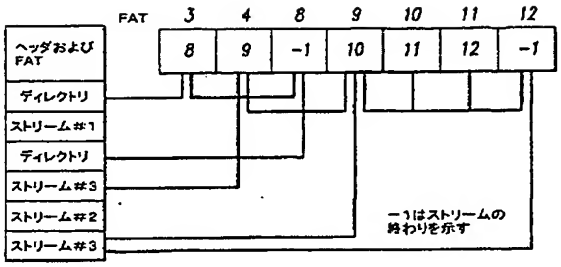


FIG.3

【図４】

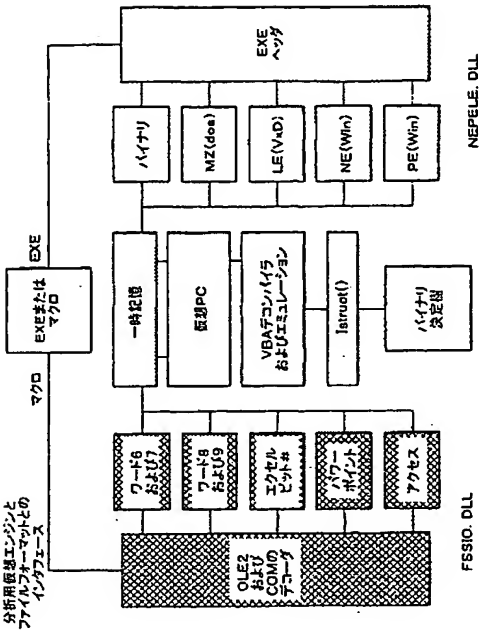


FIG.4

【図５】

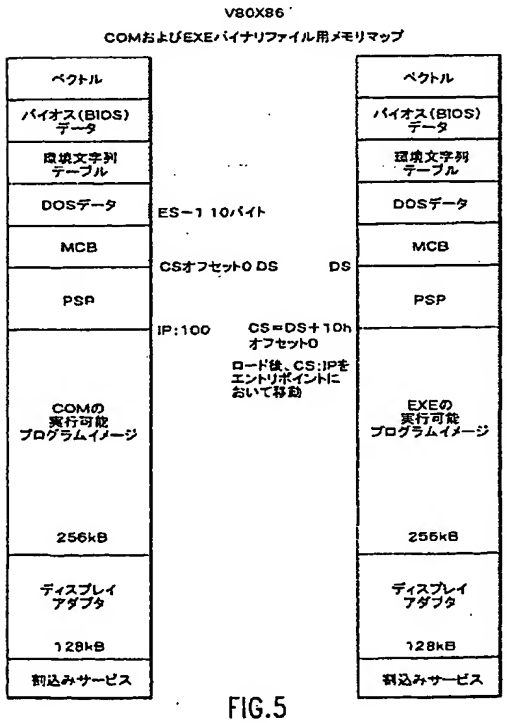


FIG.5

【図６】

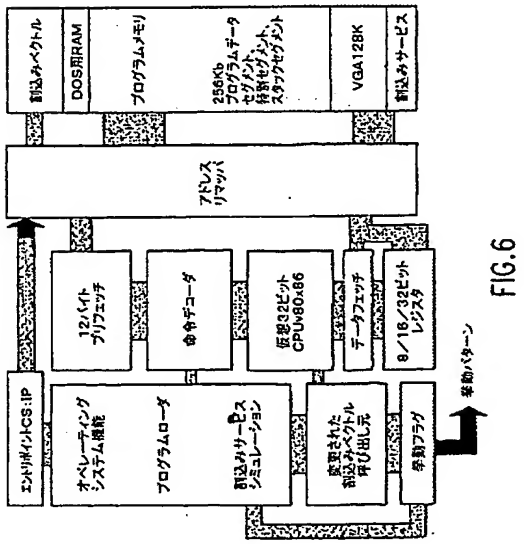


FIG.6